Пояснения по финальной задаче AI Хакатона

Мотивация

Настоящий специалист по анализу данных может работать с данными любого вида: базами банковских платежей, журналами телефонных звонков, показателями датчиков на производстве. Достаточно лишь базового представления о специфике предметной области.

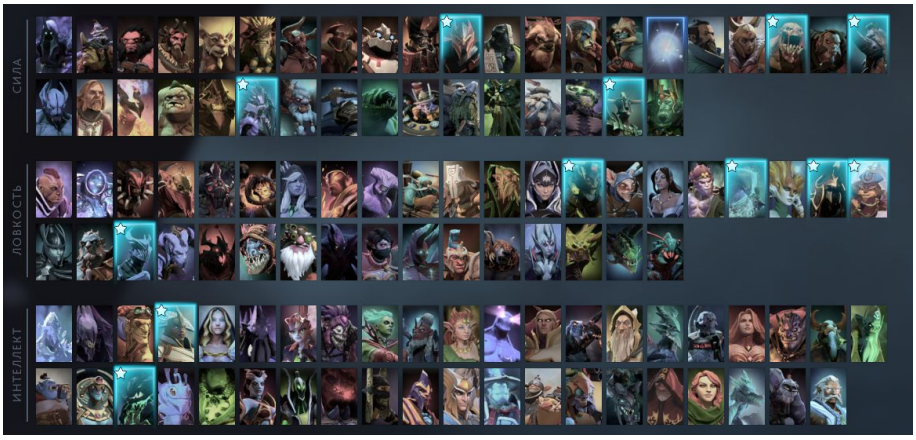
Для хакатона AI Academy мы выбрали киберспортивную дисциплину Dota 2, потому что она знакома представителям современной молодежи, многие даже являются настоящими экспертами в ней. Киберспорт — тоже спорт, поэтому задачу и идеи решений, полученные на хакатоне, можно обобщать на любые виды спорта, например, на футбол. Данные из игры несут в себе очень много полезной и интересной информации представленной в различных видах, часто встречающихся на практике: числовые и категориальные признаки, списки событий, временные ряды и прочие. Научившись работать с игровыми данными и извлекать из них полезные закономерности и знания, вы получите бесценный опыт, который в будущем можно будет применить в науке и индустрии.

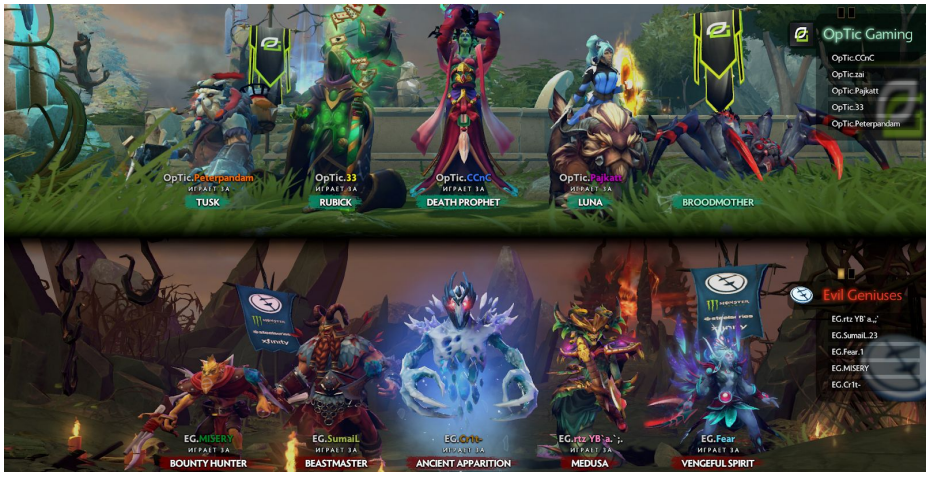
Введение в Dota 2

Команда Radiant (свет) против команды Dire (тьма). В каждой команде — ровно 5 игроков, каждый управляет своим героем.

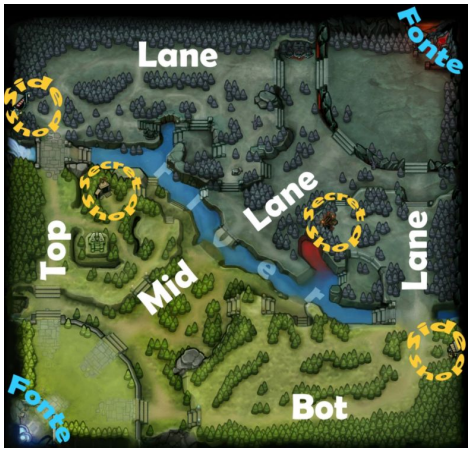
Перед началом матча каждый игрок выбирает себе героя.

На сегодняшний день в Dota 2 доступно 115 героев, каждый наделен своими способностями, у каждого есть ряд сильных и слабых сторон. Выбранная комбинация героев значительно определяет стиль игры. Существуют связки героев, которые в союзе друг с другом становятся сокрушительной силой, также может быть наоборот — неудачные связки, когда недостатки одного героя усугубляются недостатками другого.





Матч происходит на карте, на которой есть базы обеих команд, три линии, зона леса, магазины, “яма Рошана” и прочие элементы.



Во время матча игроки развивают своих героев, убивают монстров (creeps), убивают героев команды противника, разрушают башни, прокачивают свои умения, покупают в магазине предметы.

Цель команды — разрушить трон противника, кто это делает первым — тот выигрывает, ничьей не бывает.

Постановка задачи Хакатона

Матч остановлен в некоторый (произвольный) момент времени. Вам дана информация о всем происходящем до остановки, в том числе состояние всех героев. Необходимо оценить вероятность победы команды Radiant.

Обратите внимание — на этот раз не “угадать” кто победит. Угадывать — это дело неблагодарное и не научное. Нам важно оценить “шансы” на победу у команды Radiant.

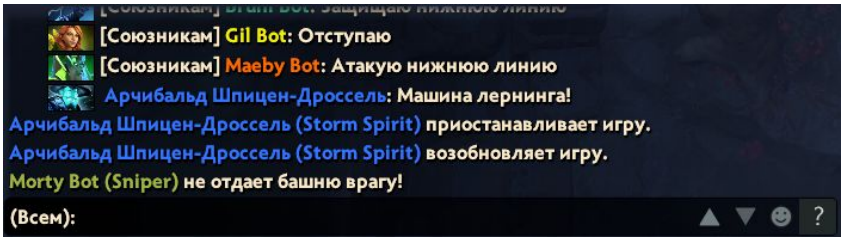
Метрика качества: ROC-AUC

В данной задаче Вам предлагается использовать площадь под ROC-кривой для оценки качества моделей. Напомним, что площадь под кривой также называют ROC-AUC (от Area Under Curve). Она принимает значения в диапазоне от 0 до 1. Чем выше ROC-AUC — тем лучше качество модели. Детальное описание ROC-AUC Вы найдете в конце документа.

Детали игры и описание данных

Участникам предоставляются сырые данные, в которых представлено детальное описание состояния матча на момент времени останова, данные закодированы в формате JSON. В объектах описания матча есть поле `players` — массив с описанием всех игроков, первые 5 элементов — игроки команды Radiant, последние 5 — игроки команды Dire.

Чат. В игре доступен общий чат, через который команды могут общаться друг с другом — желать успеха, дразниться или сообщать о технических неполадках. В поле `chat` записаны все сообщения.



Характеристики героя — определяют силу героя, его экономику и возможности по использованию заклинаний:

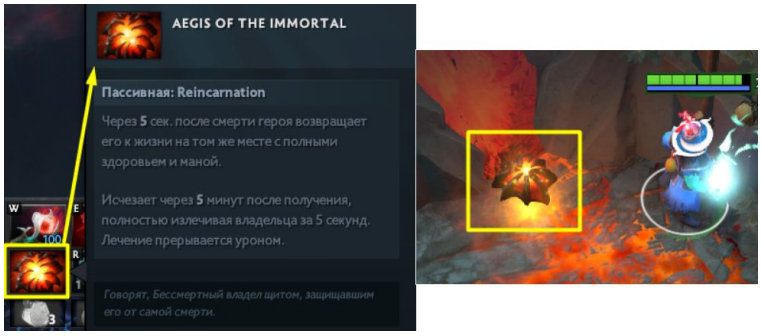
- здоровье — текущее и максимально возможное (`health, max_health`)
- мана — текущая и максимально возможная (`mana, max_mana`)
- очки опыта (`xp`)
- число добитых крипов (`lh` / `last_hits`)
- число удержанных крипов (`denies`)
- золото (`gold`)
- уровень (`level`)

Умения героев и их прокачка. Каждый герой наделен умениями, которые можно улучшать каждый раз, когда герой повышает свой уровень (поле `level`).

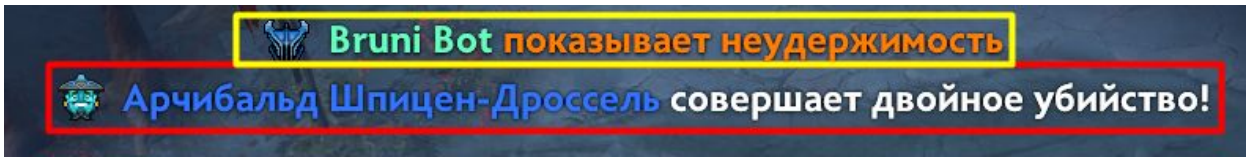
Инвентарь героя. За золото игрок может покупать игровые предметы, которые меняют характеристики и дают дополнительные умения. В данных имеется журнал всех покупок с указанием идентификатора предмета и времени покупки (поле `purchase_log`). В поле `hero_inventory` указан список текущих предметов игрока. Некоторые предметы имеют задержку между использованиями, в поле `cooldown` указано, сколько времени осталось до конца задержки. В поле `item_uses` указана статистика использования предметов игроком.

Рошан — это самый сильный нейтральный монстр в Dota 2. Он с легкостью одолевает большинство героев один на один. Игроки обычно ждут поздней фазы игры, когда фарм-герои соберут свои лучшие предметы, или пробуют убить его всей командой. После каждой смерти из Рошана всегда выпадает предмет  “Aegis of the Immortal”, его нельзя передать другому игроку, он дает владельцу вторую жизнь, реинкарнируя после смерти с задержкой в 5 секунд на месте смерти.

В поле игрока `roshans_killed` указано число убитых игроков Рошанов, а в `hero_inventory` можно обнаружить наличие Aegis у игрока.



Сообщения о событиях выдается на экране у всех игроков, когда происходит что-то переломное в игре, например разрушение здания (башни, бараков), или убийство Рошана. В поле `objectives` приведен список всех событий, которые успели произойти к моменту остановки игры.



Драка — событие в процессе игры, когда несколько героев из разных команд встречаются в одном месте и начинают бить друг друга всеми доступными умениями и заклинаниями. В списке `teamfights` доступны описания всех драк, которые произошли за игру, статистика по всем игрокам во время драки — нанесенный и полученный урон, использованные умения и т.д.

Смерть героя — не конец матча для игрока. Умирая, он теряет золото и попадает в таверну, где не может ничего делать. Через некоторое время он вернется на базу. Есть возможность ускорить возвращение — использовать “выкуп”, список всех выкупов игрока указан в поле `buyback_log`.

Статистика игрока. Игрок, совершая различные действия, пополняет статистику, которая записана в различных полях:

- число убитых героев (`kills`)
- число смертей (`deaths`)
- число помощей в драке (`assists`)
- суммарное время оглушения вражеских героев (`stuns`)
- по кому сколько нанесено урона (`damage`)
- от кого был получен урон (`damage_taken`)

Журнал событий. В полях игрока, имеющих суффикс `_log` записаны журналы различных событий.

- obs_log, sen_log — время установки “наблюдателей” двух разных видов, это специальные предметы, дающие команде обзор на карте
- purchase_log — покупки предметов в магазине
- kills_log — убийства игрока
- buyback_log — выкупы
- runes_log — взятия руны, бонуса, который появляется на карте в специальных местах каждые несколько минут

Временные ряды. Изменение базовых характеристик игрока во времени. В поле `times` указан массив из чисел — отметки по времени (в секундах), а в полях имеющих суффикс `_t` указаны значения временных рядов по соответствующим отсечкам времени:

- gold_t — изменение золота
- xp_t — изменение очков опыта
- lh_t — изменение числа добитых крипов
- dn_t — изменение числа удержанных крипов

Подробно про ROC-AUC

В задаче он-лайн тура с драками мы предсказывали факт победы или поражения команды Radiant в драке, соответственно в ответе указывали “1”, либо “0”, а качество наших предсказаний измеряли с помощью метрики Accuracy (Точность).
Как и в предыдущей задаче, целевая переменная имеет два значения “0” (False) - команда Radiant проиграла в матче, “1” (True) - команда Radiant победила в матче. Но задача теперь заключается в том, чтобы оценить шанс (число из промежутка [0, 1]) того, что команда Radiant победит в матче. На практике это означает, что в ответе теперь указывается не просто “1”, если по нашим оценкам в матче победит команда Radiant, а некоторое число близкое к 1, например: 0.9742 - шансы на победу у Radiant. Это означает, что если бы нам удалось наблюдать аналогичную ситуацию в матче 10000 раз, то в 9742 случаях действительно победила бы команда Radiant. Аналогично, если по нашим оценкам в матче должна победить команда Dire, то вместо “0” в ответ записывается некоторое число, например 0.0024. Стоит также заметить, что знакомые нам алгоритмы Logistic Regression, Decision Tree, Random Forest и т.д. на самом деле оценивают шанс события, т.е. возвращают число p из интервала [0, 1]. Далее они сравнивают число p с порогом отсечения (threshold), который по умолчанию равен 0.5, если p > threshold, то в качестве ответа возвращается “1”, в противном случае “0”.

Рассмотрим простой пример из 5 матчей. Пусть есть некоторая модель, которая предсказала следующее

Номер матча	y_prob1	y

	(Предсказанный шанс победы команды Radiant)	(Победила ли команда Radiant на самом деле?)
1	0.01	0
2	0.65	0
3	0.75	1
4	0.42	0
5	0.97	1

Пусть классификатор использует стандартный порог threshold = 0.5. Тогда имеем следующие предсказания.

Номер матча	y_prob1 (Предсказанный шанс победы команды Radiant)	y_pred (Предсказание)	y (Победила ли команда Radiant на самом деле?)	Верно ли предсказание?
1	0.01	0	0	Да
2	0.65	1	1	Да
3	0.75	1	1	Да
4	0.55	1	0	Нет
5	0.97	1	1	Да

Accuracy = 0.9, т.е. в классификатор вернул правильный ответ в 90% случаев. Взглянем теперь на картину более детально. Чтобы лучше оценить качество модели, нужно знать её слабые и сильные стороны.

Давайте посчитаем число матчей, когда классификатор вернул “1”, хотя на самом деле случилось событие “0” и обозначим эту величину FP (от False Positives - число событий, когда модель неверно предсказала “1” - положительное событие, также такие случаи называют “ложными срабатываниями”). В нашем случае FP = 1, потому что в 4-ом матче модель оценила шанс события “1” в 0.55, т.к. 0.55 > threshold, то в качестве ответа имеем y_pred = 1. Ясно, что FP = 0, когда для всех событий с y = 0 модель предсказывает y_pred = 0. На практике очень удобно работать с величинами, которые принимают свои значения в диапазоне от 0 до 1. Рассмотрим вместо количества ложных срабатываний FP их долю среди событий с y = 0, обозначим эту величину FPR (от False Positives Rate - доля ложноположительных ответов).

$FPR = \frac{FP}{N}$, где N - число событий с фактическим значением y = 0 (Negatives). Стоит заметить, что FPR принимает значения в диапазоне от 0 до 1. В нашей задаче $FPR = \frac{1}{2} = 0.5,$ т.е, доля ложноположительных событий составляет 50%!

Аналогичным образом можно определить число верно положительных предсказаний TP (от True Positives), как число матчей, когда случилось событие “1” и классификатор вернул ответ “1”. В нашем примере TP = 3. Доля верно положительных предсказаний среди всех событий с фактическим значением y = 1 будем обозначать TPR.

$TPR = \frac{TP}{P}$, где P - число событий с фактическим значением y = 1 (Positives). В нашем примере $TPR = \frac{3}{3} = 1.$

Обычно к этому моменту назревает естественный вопрос: “Зачем мы мучались и вводили величины TPR и FPR, когда есть вполне понятная метрика Accuracy?” Чтобы продемонстрировать ответ на него, давайте рассмотрим уже известную задачу с пятью матчами. Никто не запрещал нам выбрать порог отсечения threshold отличным от 0.5. Отлично! Пусть теперь будет порог отсечения threshold = 0.75.

Номер матча	y_prob1 (Предсказанный шанс победы команды Radiant)	y_pred (Предсказание)	y (Победила ли команда Radiant на самом деле?)	Верно ли предсказание?
1	0.01	0	0	Да
2	0.65	0	1	Нет
3	0.75	1	1	Да
4	0.55	0	0	Да
5	0.97	1	1	Да

Теперь в матче 2 модель предсказывает событие “0”, потому что 0.65 < threshold, а в матче 4 предсказывается событие “0”. Accuracy модели по прежнему 0.9, но ошибку она совершает теперь не в матче 4, а в матче 2. Изменились ли при этом FPR и TPR?

$FPR = \frac{0}{2} = 0$, $TPR = \frac{2}{3} \sim 0.66.$ Действительно, значения метрик изменились. Теперь не встречается ошибка, когда модель “называет нулик единичкой” (ложное срабатывание), но встречается ошибка, когда модель “называет единичку нуликом”.

Вот мы и подошли к ответу на поставленный вопрос: “Зачем мы мучались и вводили величины TPR и FPR, когда есть вполне понятная метрика Accuracy?”

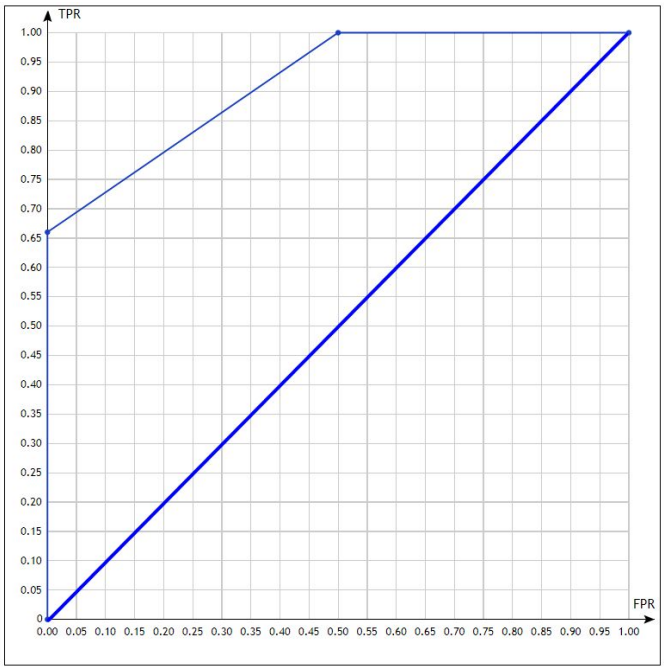
Ответ: Существуют различные типы ошибок модели: “нулики называются единичками”, “единички называются куликами”, при этом доли этих ошибок зависят от порога отсечения threshold.

Следующий вопрос: “Что же все-таки за ROC-кривая и зачем мерить под ней площадь?”

Вернемся к нашей задаче с 5 матчами и посчитаем TPR, FPR еще для нескольких порогов отсечения. Результат запишем в таблицу:

threshold	FPR	TPR
1	0	0
0.75	0	0.66
0.5	0.5	1
0	1	1

Возьмем теперь значения из столбцов FPR, TPR и нарисуем их напротив друг друга. Получим график, проходящий через точки: (0; 0) (0; 0,66) (0,5; 1) (1; 1). Этот график и будет называться ROC-кривой.



Заметим, что диагональ квадрата получается при FPR = TPR для любых значений threshold. Такая ситуация возникает, когда в качестве модели выступает монетка. Каждый раз, когда нужно предсказать “0” или “1”, подбрасывается монетка, в зависимости от того, выпала ли решка или орел, модель предсказывает соответственно “0” или “1”. В этом случае “нули” и “единички” будут так сильно перемешаны между собой, что FPR = TPR = 0.5. Модель случайного угадывания на графике ROC-кривой используют в качестве

baseline - опоры, относительно которой можно измерять качество новых моделей.
Площадь под кривой обозначают AUC (Area under curve).

ROC-AUC(baseline) = 0.5
ROC-AUC(model) = 0.83

ROC-AUC равен доле пар объектов вида (объект “1”, объект “0”), которые алгоритм верно упорядочил, т.е. первый объект идёт в упорядоченном списке раньше. Чем больше площадь под ROC-кривой, тем выше ROC-AUC, тем лучше алгоритм упорядочивает ответы, тем точнее классификация.